Journal of Population Therapeutics & Clinical Pharmacology

RESEARCH ARTICLE
DOI: 10.47750/jptcp.2023.30.12.004

Improving Task Scheduling on The Cloud Through Low Latency and Cost Effective Technique

Bennet Praba M.S[1*], Sahil Saju[2], Gokul N[3], Karthick raja G[4]
[1,2,3,4]Computer Science and Engineering, SRMIST, Ramapuram, Chennai, India
***Corresponding author:** Bennet Praba M.S, Computer Science and Engineering, SRMIST, Ramapuram, Chennai, India

Submitted: 15 March 2023; Accepted: 17 April 2023; Published: 04 May 2023

ABSTRACT

Offloading is emerging as a promising idea to allow handheld devices to access intensive applications without incurring performance or power costs. This is especially useful for businesses that want to run line-of-business applications on handhelds.. However, developing applications using cloud computing resources is a challenge because supporting the low-latency and scalability needs of applications requires highly dynamic orchestration of heterogeneous resources at different levels of the network hierarchy. It Is difficult. To reduce this complexity, this application provides a programming model that provides simplified programming abstractions and supports applications that scale dynamically at runtime. The goal of offloading is to decide for or against offloading. Code offload can make decisions in a number of ways. Much of the research on code swapping focuses on more sophisticated if-else conditions. Code offloading machine learning is currently an important research topic. Machine learning methods are employed to significantly improve the currently recommended methods and to perform robustly when tackling a wide range of learning tasks

Keywords: *Offloading, run line-of business, heterogeneous, orchestrating highly, sophisticated, robustly whilst tackling*

INTRODUCTION

In recent years, the popularity of handheld devices has increased the demand for mobile applications that offer high performance and power efficiency. Offloading has emerged as a promising approach to address these needs by allowing resource-intensive tasks to be performed on remote cloud resources. However, to make offloading profitable for the company, he must address two key challenges. Ensure data protection and provide offload at scale with a variety of handheld target and computing resource capabilities.

Additionally, developing cloud-based applications involves complex orchestration of heterogeneous resources at various network layers, which can pose significant challenges in achieving low latency and scalability. To address these issues, this white paper presents a programming model that supports dynamic runtime scaling and provides simplified programming abstractions. The goal of this research is to enable timely task execution while maintaining quality of service (QoS), maximizing performance while minimizing energy consumption in cloud data centers.

Machine learning techniques are also being explored to improve offload decision-making processes, with the aim of achieving robust and reliable results across a wide range of learning tasks. Ultimately, the purpose of this research is to provide a cost-effective solution for reliable, low-latency task scheduling in cloud environments. This makes offloading a more practical and accessible approach for businesses. Mobile Edge Computing (MEC) is one of the most promising technologies for next-generation wireless communication systems. This article examines the problems of dynamic caching, computational offloading, and resource allocation in cache-based multi-user MEC systems with stochastic task arrivals. There are some computationally intensive tasks in the system and each mobile user (MU) needs to perform the tasks locally or remotely on one or more MEC servers by offloading the task data. Commonly used tasks can be cached on her MEC server to avoid duplication of work when outsourced. Cloud Radio Access Network (C-RAN) has emerged as a potential next-generation access network technology candidate to handle the growing mobile data traffic, and Mobile Cloud Computing (MCC) has emerged as a resource-constrained candidate. emerging as a possible candidate.

It offers a promising solution for some mobile users to perform calculations. strenuous task. This document introduces C-RAN with MCC, leveraging the above two cloud-based techniques to improve both performance and power efficiency. An interesting approach to running resource-intensive applications in mobile cloud computing environments is to offload computation and data to the cloud. However, offloading to a remote cloud is not always the best solution due to the increased latency and power consumption associated with offloading and intermittent wireless communication. Our main goal was to use this concept to optimally decompose the set of tasks performed by the mobile client and the two-tier cloud architecture. One is for individual users and the other is a collaborative mobile application.

A comparative study was performed to evaluate the performance of the proposed TS-DT algorithm. It runs under existing algorithms. Heterogeneous Early End Time (HEFT), a priority technique Similarity to the ideal solution incorporating the entropy weighting method (TOPSIS-EWM) and combined with Q-learning Heterogeneous earliest finish time (QL-HEFT). Our results show that the proposed TS-DT algorithm outperforms existing ones. HEFT, TOPSIS-EWM, and QL-HEFT algorithms by reducing manufacturing margins by 5.21%, 2.54%, and 3.32%, respectively, and improving resources Utilization improved by 4.69%, 6.81%, and 8.27%, respectively, and load balancing improved by 33.36%, 19.69%, and 59.06%.average.

Reasons To Pursue This Project

Optimizing Task Scheduling

The task scheduling process aims to determine the optimal sequence of job execution that utilizes the least amount of resources, including time, processing power, memory, and other resources. By optimizing task scheduling, organizations can improve resource utilization and reduce costs while ensuring that all tasks are completed on time and according to specific requirements.

To achieve this goal, the task scheduling process involves evaluating various factors, including task execution time, the cost of execution, and the availability of resources. By considering these factors, organizations can determine the optimal sequence of job execution that minimizes total execution time and cost while ensuring proper resource utilization.

Reducing Cost Efficiency

In addition to improving cost efficiency, optimizing task scheduling can also help organizations meet specific requirements and constraints related to task execution, such as deadlines, dependencies, and priority levels. By balancing these factors, organizations can achieve optimal task scheduling that meets their unique needs and requirements.

Previous Work

This document [1] introduces the Task Scheduling Decision Tree (TS-DT) algorithm, a decision tree-based task scheduling algorithm. The performance of the proposed task scheduler is evaluated against load balancing parameters such as makespan, resource utilization, and power consumption in a heterogeneous environment. The work in this white paper makes the following important contributions. They proposed a new task scheduling algorithm based on decision trees for scientific workflows.

The goal is to minimize [2]computing and network energy.

Strict minimum streaming rate and maximum computing power, and the entire ecosystem under network constraints means. to this end: the energy of the target ecosystem, The virtualized and multi-core nature of fog/cloud servers. the resulting problem is not convex, Since both continuous and discrete variables are included, the optimality-preserving decomposition is A cascade of (continuous) resource assignment subproblems and (discrete) task assignment subproblems. we Solve the first subproblem numerically by a series of well-designed gradient-based adaptive iterations,We address the second sub-problem by relying on ad-hoc developed elite genetics algorithm.

Mobile Edge Computing [3] (MEC) has become a key technology to offload the computational burden of SMDs.

Reduce service latency for compute-intensive applications. Benefit from network capabilities Virtualization enables MEC to compute and integrate Cloud Radio Access Network (C-RAN) on UDN. communication cooperation. However, this is the case for stochastic arrival times of computational tasks and time-varying channel conditions.

The challenge of outsourcing computational tasks online using energy efficient computing and wireless resources

management. This article examines MEC-aware task offload and resource allocation issues. High-density C-RAN aimed at optimizing network energy efficiency.

The Industrial Internet of [4] Things (IIoT) offers unprecedented opportunities to drive manufacturing intelligence and smartness. However, timely processing of large-scale IIoT data by traditional computational frameworks such as cloud computing is non-trivial due to costly resource consumption, unacceptable delays, and unacceptable loads on backbones. Mobile Device Cloud (MDC) leverages the idle resources of intelligent objects at the edge, and its flexible resource deployment and near task offload make it promising for IIoT data analytics.

[5]. This white paper first introduces a new mobile cloud called Opportunistic Task Scheduling over Co-located Clouds (OSCC) that provides flexible cost/latency trade-offs between traditional remote cloud service mode and mobile cloudlet service mode. Suggest a let-based service mode. We then conduct a detailed analytical study of OSCC modes and solve the power minimization problem by trade-offs between remote cloud mode, mobile cloudlet mode, and OSCC mode

[6] Mobile Edge Computing (MEC) is one of the most promising technologies for next-generation wireless communications. Communications system. This article discusses dynamic caching, computation offloading, and Resource Allocation in a Cache-Assisted Multiuser MEC System with Stochastic Task Arrival. there are some Computationally intensive tasks in the system, and each mobile user (MU) must perform tasks either locally or not. Run remotely on one or more MEC Servers by outsourcing task data. Popular tasks can be cached on his MEC server Avoid duplication when outsourcing.

In this paper,[7] a cloudlet-based task-centric model is proposed. The system is first presented with analysis and discussion Local Mobile Devices, Cloudlets, Remote Clouds, Network Roles and Core Perspectives function. Charging scenarios are also formulated in proposed systems, including and task module, compute or network module, and joint load planning proposed algorithm.

This white paper proposed [8] a new framework for modeling mobile applications as location-time workflows.

The point here is that this abstraction models the usage patterns of mobile user services based on user mobility. our

The main goal was to use this concept to optimally decompose the set of tasks performed on the mobile client.

Two-tier cloud architecture for two types of mobile applications: single-user and single-user Collaborative mobile application.

[9]. A new C-RAN architecture including mobile clones is proposed in this white paper. Two cloud-based techniques. In particular, I assume that the task should be performed on mobile devices We clone each UE and model this task with two functions. H. Total number of CPU cycles required for this Total data size required to complete this task and send the results to the UE via C-RAN. We Modeling this problem minimizes the overall energy cost of mobile clouds and mobile networks. Optimization problem when acquiring QoS.

in this paper, [10] we proposed a SOBDO that optimally selects targets to load resource-intensive tasks. Between different targets in an MCC environment. SOBDO uses resource-intensive static partitioning Applied to smaller tasks, apply the concepts of auction theory to select the best target for your cargo special mission. The SOBDO testbed evaluation compares the efficiency of different communication modes.3G mobile communication, WiFi.

Proposed System

In this Paper, we introduce the TS-DT algorithm to reduce the makespan, enhance load balance, and maximize utilization of the resource. The algorithm consists of three phases: the priority task, the resource matrix, and the resource allocation phase. First, the task priority phase is used to assign a rank for each task. The resource matrix phase is used in collecting the tasks' features in the form of a matrix, while the resource allocation phase is where tasks are scheduled on the proper VMs using the decision tree.

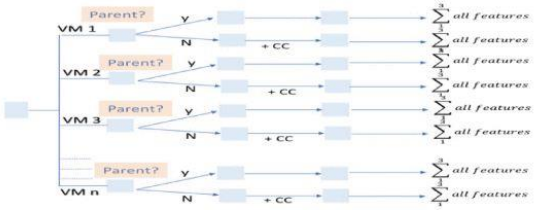

FIG 1: The Proposed TS-DT Algorithm Using Decision tree

Ts-Dt Algorithm

A decision tree is a hierarchical data structure that uses the divide-and-conquer method to represent data. A decision tree, on the other hand, is a rooted tree with leaf and non-leaf nodes. The decision criteria for classification and regression trees obviously depend on the decision tree. A decision tree, on the other hand, is a rooted tree with leaf and non-leaf nodes. Leaf nodes represent classifications or decisions, and non-leaf nodes represent choices, dividing the instance space into two or more subspecies based on discrete functions of input attribute values.

The decision tree algorithm for task scheduling can be expressed as:

$T(i,j) = \min\{\max\{T(i-1,k), C(k+1,j)\}$

where T(i,j) represents the minimum time required to complete tasks i through j and C(i,j) represents the cost of completing tasks i through j .

$C(i,j) = C\_comp(i,j) + C\_storage(i,j) + C\_network(i,j)$

where C_comp(i,j) represents the compute cost associated with executing the task from i to j, C_storage(i,j) represents the storage cost associated with the task, and C_network(i,j) represents the network cost associated with the task.

$R(i,j) = R\_compute(i,j) + R\_storage(i,j) + R\_network(i,j)$

where R_compute(i,j) represents the availability of compute resources for task execution, R_storage(i,j) represents the availability of storage resources, and R_network(i,j) represents the availability of network resources.

By considering these values and equations, the TS-DT algorithm can effectively schedule tasks

in cloud computing environments, minimizing the total execution time and cost while ensuring resource availability and meeting other constraints.

This algorithm aims to minimize the total time required to complete a set of tasks subject to various constraints.

The task scheduling decision tree algorithm is the process of determining the optimal order in which tasks should be completed based on their priority and estimated time. The algorithm involves building a decision tree, assigning probabilities to each decision, calculating expected values for each branch, and choosing the best branch. This process is repeated for each decision point until the complete schedule is created. This algorithm is effective at optimizing task schedules, but requires accurate estimates and a good understanding of project requirements and dependencies.

A task scheduling decision tree algorithm is a type of decision-making process used to optimize the scheduling of tasks within a project. The algorithm evaluates the expected value of each branch of the decision tree to determine the optimal order for completing the task.

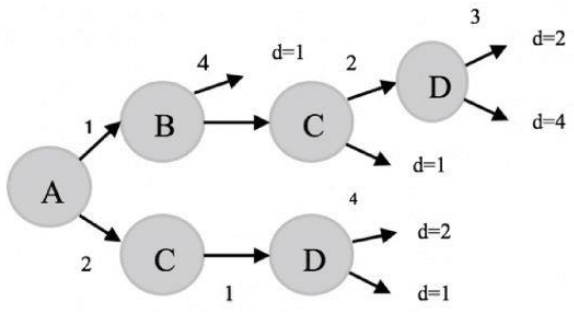

FIG 2: Decision Tree Structure

The first step in the algorithm is to prioritize each task by assigning a score based on factors such as deadline and importance to the overall project. The estimated time required to complete each task is determined using historical data or expert opinion. A decision tree is then built for each task, showing possible outcomes at each decision point. For example, if your first task has three dependencies, create three branches for each possible order in which those dependencies can be completed. Each decision is then assigned a probability based on the estimated time required for each task and other relevant factors.

Architecture Diagram

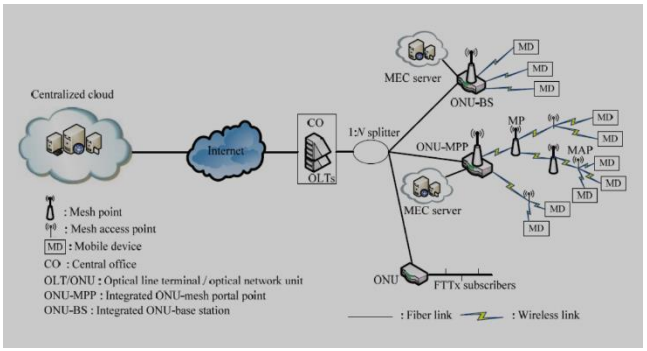

FIG 3

Module Description

Module 1: code profiler

Offloading is the opportunistic process of relying on a remote server to execute delegated code from a mobile device. In the process, the mobile device is given local decision logic to detect resource-intensive parts of the code.

Network communication can presume mobile phones with less computation required to execute code. Code Profiler decides what to swap out. So the method, thread, or class portion of the code (C) is identified as follows: Outsourcing Candidate (OC). Code splitting requires you to choose which code to swap out. Code can be split Through various strategies; for example, software developers can explicitly choose which code to swap out. Special static annotations (@Offloadable, @Remote, etc.). Other strategies implicitly parse the code at runtime. Automated mechanism So when the application is installed on the device, the mechanism chooses which code to offload. Mechanisms implement strategies such as static analysis to deduce whether a piece of code is intensive and traces of history. Automated mechanisms are preferred over static mechanisms because they allow customization of the code that is executed different devices.

Module 2: system profiler

System Profiler is responsible for monitoring several parameters of your smartphone. B. Available bandwidth and data size Power to send and run code. These parameters affect when to outsource to the cloud. Conceptually The offload process is optional and the effort required to run her OC on a mobile device When called remotely as local execution. Otherwise discharge is not recommended as an excessive expenditure of energy and time Consumed when transferring data to the cloud.

Module 3: Decision engine

A decision engine is an inference engine that suggests when to outsource to the cloud. The engine gets the data received from the system and code profilers and apply some logic to them (linear programming, fuzzy logic, Markov chains, etc.)

The engine can measure whether handsets are getting tangible benefits from outsourcing to the cloud. the engine

If the result is positive, the swapping mechanism is enabled and the code is called remotely. otherwise the code is executed local. Mobile phones are offloaded to the cloud at a rate dependent on the size of the data and available bandwidth. If code offloading is counterproductive for the device, it's usually due to an imprecise wrong reasoning process Based on the range of observable parameters that System Profiler can monitor The proposed system fully supports a fully functioning decision engine.

The engine uses the past task execution to predict their execution time and output size. The engine along with the network profiler estimates the running time of any given task in both local and cloud contexts. Predictions are possible only past records are exists. It is not the case for the freshly installed applications. The overcome this limitation, the Proposed system offers two execution modes:

1.Concurrent Mode

2.Optimistic Mode

In concurrent mode, tasks are simultaneously run on both local and cloud contexts. Alternatively, in optimistic mode, the task is run on only one context depending on the output of the decision engine. Tasks are always run concurrently when no past records are available: this also allows us to feed data into the decision engine.

Performance Evaluation

According to the implementation results, it is found that the proposed TS-DT algorithm outperforms the HEFT, TOPSIS-EWM, and QL-HEFT algorithms. This is because of the following reasons:

During the Task Priority phase, the proposed TS-DT algorithm determines the proper task to be executed by increasing its priority while using the task length and number of childs. During the Resource Allocation phase, the decision tree and the summation of the features in the task's matrix are used to select the VM with the lowest value. Some features are used to enhance the make span (i.e., computation cost, Earliest Finish Time (EFT), and parent location).

RESULT

1. According to the comparative results in Table 4, We found that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 5.21%, 2.54%, and 3.32%, respectively.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	-4.0900008	-0.2440566	-0.4378286
10	-13.106437	-8.647954	-9.9941048
20	-1.7497734	-0.7443627	-1.3288819
40	-1.9063134	-0.5330695	-1.5379343

FIG 4: Improved Average Rate of Makespan (in %) of the Proposed TS-DT Algorithm

2. According to our comparative results in Table 5 show that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by

approximately 4.69%, 6.81%, and 8.27%, respectively.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	1.62271401	3.73324816	4.27320777
10	8.20109244	10.1293424	9.2128822
20	5.25282666	7.85525259	9.63403989
40	3.67758441	5.52411861	9.95996476

FIG 5: Improved Average Rate of Resource Utilization (in %) for the Proposed TS-DT Algorithm

3. According to the comparative results in Table 6, it is found that the proposed TS-DT algorithm outperforms, in average, the default HEFT, TOPSIS-EWM, and QL-HEFT algorithms by approximately 33.36%, 19.69%, and 59.06%, respectively.

No. of VMs	HEFT	TOPSIS-EWM	QL-HEFT
5	30.0693209	43.158865	-24.383398
10	-51.539583	-32.022823	-69.803277
20	-56.820015	-44.696929	-74.803327
40	-55.15023	-45.207672	-67.241872

FIG 6: Improved Average Rate of Load Balance (in %) of TS-DT Algorithm

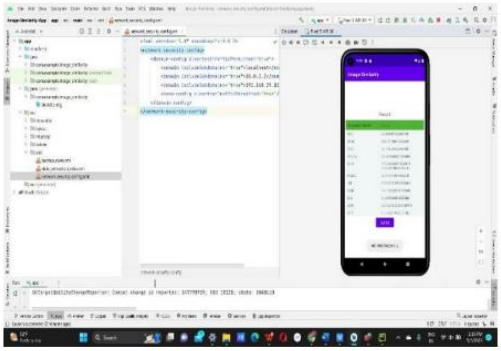

FIG 7: Output

CONCLUSION AND FUTURE WORK

In this system, we proposed a framework for partial computational unloading to determine energy-optimal datasets. I entrust it to Mex. Application and network parameters heavily influence and are taken into account in outsourcing decisions taken into account by the proposed framework. This work includes all major energy consumption units. Radio codec and local Calculate your units to get a more accurate partial offload configuration. Energy efficiency during use is also evaluated Compression to reduce transmitted data. The use of compression is observed despite the additional energy consumption As such, it is a power saving option for his MEC applications which are computationally intensive. Regarding the obtained energy-optimal discharge configuration using the proposed framework.

Our results show that the proposed TS-DT algorithm outperforms the existing HEFT, TOPSIS-EWM, and QL-HEFT algorithms by reducing make span by 5.21%, 2.54%, and 3.32%, respectively, improving resource utilization by 4.69%, 6.81%, and 8.27%, respectively, and improving load balancing by 33.36%, 19.69%, and 59.06%, respectively in average.

Future work

As a future direction, we would like to explore algorithms for more advanced compute offload scenarios. It may contain multiple subtasks that have dependencies on each other and may need to be run on a regular basis.

REFERENCES

1. Multiobjective Task Scheduling in Cloud
2. Environment Using Decision Tree Algorithm
3. Hadeer Mahmoud 1,2, Mostafa Thabet3, Mohamed H. Khafagy1, And Fatma A. Omara4, (Member, Ieee).
4. EcoMobiFogDesign and Dynamic Optimization of a 5G Mobile-Fog-Cloud Multi-Tier Ecosystem for the Real-Time Distributed Execution of Stream Applications Enzo Baccarelli , Michele Scarpiniti and Alireza Momenzadeh 2019.
5. Dynamic Task Offloading and Resource Allocation for Mobile-Edge Computing in Dense Cloud RAN D Qi Zhang , Lin Gui , Fen Hou , Jiacheng Chen , Shichao Zhu and Feng Tian 2020.
6. BC-Mobile Device Cloud:A Blockchain-Based Decentralized Truthful Framework for Mobile Device BC Cloud Mu Wang , Changqiao Xu ,

Xingyan Chen , Lujie Zhong , Zhonghui Wu and Dapeng Oliver Wu 2021.

7. Opportunistic Task Scheduling over Co-Located Clouds in Mobile Environment Min Chen , Yixue Hao , Chin-Feng Lai , Di Wu , Yong Li and Kai Hwang 2018.
8. Deep reinforcement learning for dynamic computation offloading and resource allocation in cache- Dassisted mobile edge computing systems Samrat Nath and Jingxian Wu 2020.
9. A Task-Centric Mobile Cloud-Based System to Enable Energy-Aware Efficient Offloading Azzedine Boukerche , Shichao Guan and Robson Eduardo De Grande 2018.
10. On Optimal and Fair Service Allocation in Mobile Cloud Computing M. Reza Rahimi , Nalini Venkatasubramanian , Sharad Mehrotra and Athanasios V. Vasilakos 2018.
11. Joint Energy Minimization and Resource Allocation in C-RAN with Mobile Cloud J Kezhi Wang , Kun Yang and Chathura Sarathchandra Magurawalage 2018.
12. Auction-Based Optimal Task Offloading in Mobile Cloud Computing Sudip Misra , Bernd E. Wolfinger , M.P. Achuthananda , Tuhin Chakraborty , Sankar N. Das and Snigdha Das 2019.